Robotics Toolbox for MATLAB

He Jinshou, Deng Fuqin, Yang Guanglu
hjs00@126.com, dengfuqin_hitsz@126.com, glasses-21@163.com
Harbin Institute of Technology
Shenzhen Graduate School,CoME Grade 2005
Wednesday, June 07, 2006

Contents

1 Introduction

This Toolbox provides many functions that are useful in robotics including such things as Twists, Kinematics, Manipulator forward Dynamics, and Manipulator Jacobian. The Toolbox is useful for simulation as well as analyzing results from experiments with real robots.

The Toolbox is provided for the book "An Mathematical Introduction to Robotic Manipulator" (preprint, 3, Jan 1994) by Richard M.Murray, Zexiang Li and S.Shankar Sastry. Many problems in the book about homogenous transformation, twists, and kinematics can be solved by this Toolbox, and some example codes are provided. Most functions in the toolbox support symbol computation.

The Toolbox works with MATLAB version 6.5 and has been tested on a Windows XP.I don't know whether the Toolbox can work well under the older versions or other platforms.

2 Acknowledgement

Most of the functions are finished by Fuqin Deng and me. Thank Prof. Zexiang Li for his perfect lessons about robotics. Most ideas in this toolbox are in his books, we just implement it. I also learned a lot from the robotics toolbox written by Peter I. Corke(pic@cat.csiro.au http://www.cat.csiro.au/cmst/staff/pic/robot). His toolbox works very well, but doesn't support twist, which is the main idea in the book "An Mathematical Introduction to Robotic Manipulator", so I add most of the relative functions. I think twist is a very good idea for robotics. It's very simple and powerful, but few books talk about it.

3 Tutorial

3.1 Rotational Motion in R3

Let A be the inertial frame, B the body frame,and $\mathbf{x}_{ab}, \mathbf{y}_{ab}, \mathbf{z}_{ab} \in \mathbb{R}^3$ the coordinates of the principal axes of B relative to A. Stacking these coordinate vectors next to each

other, we define a 3×3 matrix:

$$R_{ab} = \begin{bmatrix} \mathbf{x}_{ab} & \mathbf{y}_{ab} & \mathbf{z}_{ab} \end{bmatrix}$$

We call a matrix constructed in this manner a rotation matrix: every rotation of the object relative to the ground corresponds to a matrix of this form.

The cross product between two vectors $a, b \in \mathbb{R}^3$ is defined as

$$a \times b = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix} \qquad 3\text{-}1$$

Since the cross product by a is a linear operator, $b \mapsto a \times b$ may be represented using a matrix. Defining

$$(a)^{\wedge} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \qquad 3\text{-}2$$

We can write $a \times b = (a)^{\wedge} b$

Functions:

function M=Rot(type, sita)

Generate a Rotation matrix, type=1 2 3 represent for the rotation axis x, y, z respectively. sita represent for the rotation angle.

In fact, this function generate a 4*4 matrix M as follows:

$$M(type, \alpha) = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & if \quad type = 1 \\ \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & if \quad type = 2 \\ \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & if \quad type = 3 \end{cases} \qquad 3\text{-}3$$

function M=Trans(dx, dy, dz)

Generate a translation matrix, dx, dy, dz represent the translation along x, y, z axis respectively.

$$M(dx,dy,dz)=\begin{bmatrix}1 & 0 & 0 & dx\\ 0 & 1 & 0 & dy\\ 0 & 0 & 1 & dz\\ 0 & 0 & 0 & 1\end{bmatrix} \quad 3\text{-}4$$

function y=VectorProduct(x1,x2)

Compute the vector product of two 3*1 vector x1, x2, returns a 3*1 vector

Reference See formula 3-1

function y=v3tor3(v)

Compute the skew symmetry matrix of a 3*1 vector v

Reference See formula 3-2

3.2 Exponential coordinates and Homogeneous representation

Define

$$e^{\hat{\omega}t}=I+\hat{\omega}t+\frac{(\hat{\omega}t)^2}{2!}+\frac{(\hat{\omega}t)^3}{3!}+\cdots \quad 3\text{-}5$$

Lemma

$$e^{\hat{\omega}\theta}=I+\frac{\hat{\omega}}{\|\omega\|}\sin(\|\omega\|\theta)+\frac{\hat{\omega}^2}{\|\omega\|^2}(1-\cos(\|\omega\|\theta)) \quad 3\text{-}6$$

We can verify that $\exp(\hat{\omega}\theta)$ is indeed a rotation matrix.

Let $q_a, q_b \in \mathbb{R}^3$ be the coordinates of a point *q* relative to frames *A* and *B*, respectively. Given *qb*, we can find *qa* by a transformation of coordinates:

$$q_a = p_{ab} + R_{ab}q_b \quad 3\text{-}7$$

The transformation of points and vectors by rigid transformations has a simple representation in terms of matrices and vectors in R4. Define a vector in R4 as

$$\bar{q}=\begin{bmatrix}q_1\\ q_2\\ q_3\\ 1\end{bmatrix} \quad 3\text{-}8$$

These are called the *homogeneous coordinates* of the point *q*.
Vectors, which are the difference of points, then have the form

$$\bar{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} \tag{3-9}$$

Using the preceding notation for points, we may represent it in linear form by writing $q_a = p_{ab} + R_{ab} q_b$ as

$$\bar{q}_a = \begin{bmatrix} q_a \\ 1 \end{bmatrix} = \begin{bmatrix} R_{ab} & p_{ab} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q_b \\ 1 \end{bmatrix} =: \bar{g}_{ab} \bar{q}_b \tag{3-10}$$

Define

$$se(3) := \{(v, \hat{\omega}) : v \in \mathbb{R}^3, \hat{\omega} \in so(3)\} \tag{3-11}$$

In homogeneous coordinates, we write an element $\hat{\xi} \in se(3)$ as

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \tag{3-12}$$

An element of *se*(3) is referred to as a *twist*, or a (infinitesimal) generator of the Euclidean group. We define the $\vee$ (vee) operator to extract the 6-dimensional vector which parameterizes a twist,

$$\begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3-13}$$

and call $\xi := (v, w)$ the *twist coordinates* of $\hat{\xi}$. The inverse operator, ^ (wedge) forms a matrix in *se*(3) out of a given vector in R6:

$$\begin{bmatrix} v \\ \omega \end{bmatrix}^{\wedge} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \tag{3-14}$$

The 4×4 matrix $\bar{g}_{ab}$ is called the *homogeneous representation* of $g_{ab} \in SE(3)$. In general, if $g = (p, R) \in SE(3)$, then

$$\bar{g} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{3-15}$$

Exponential coordinates for rigid motion and twists

Suppose the original position of the rigid body is p(0), after a rigid body transformation, the position is p(t),we can prove that

$$\bar{p}(t) = e^{\hat{\xi} t} \bar{p}(0) \tag{3-16}$$

Where $e^{\hat{\xi} t}$ is the matrix exponential of the 4×4 matrix $\hat{\xi} t$, defined (as usual) by

$$e^{\hat{\xi}t} = I + \hat{\xi}t + \frac{(\hat{\xi}t)^2}{2!} + \frac{(\hat{\xi}t)^3}{3!} + \cdots \quad 3\text{-}17$$

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \quad 3\text{-}18$$

Lemma: if the rigid body transformation is pure translational motion(ω=0), then

$$\boxed{e^{\hat{\xi}\theta} = \begin{bmatrix} I & v\theta \\ 0 & 1 \end{bmatrix} \qquad \omega = 0} \quad 3\text{-}19$$

Otherwise

$$\boxed{e^{\hat{\xi}\theta} = \begin{bmatrix} e^{\hat{\omega}\theta} & (I - e^{\hat{\omega}\theta})(\omega \times v) + \omega\omega^T v\theta \\ 0 & 1 \end{bmatrix} \qquad \omega \neq 0} \quad 3\text{-}20$$

Functions:

function y=exp3_1(omiga,sita)

Compute the exponential product. Using formula

$$e^{\omega\hat{}\,\theta} = I + w\hat{}\,\sin(\omega) + w\hat{}^{\,2}\,(1-\cos(\omega))$$

Note: omega is a 3*1 unit vector, sita is a real number denoting the angle of rotation.

This function returns a 3 by 3 matrix whose value is the exponential product $e^{\omega\hat{}\,\theta}$.

function y=screw(w, q)

Compute the pure rotation twist--a 6 by 1 vector--which rotates about the axis w(a 3*1 unit vector), and across the point q(a 3*1 vector).The function is useful in forward kinematics.

function y=exp4_1(w, sita, v)

Compute the exponential product of a twist, the twist is [v, w]',where w is a 3*1 unit vector, and v is a 3*1 vector, sita is the rotation angle, which is a real number or symbol variable.

Reference see formula 3-19 and 3-20.

function y=exp4_2(vw,sita)

Compute the exponential product of a twist vw, a 6 by 1 vector, and the rotation angle sita, which is a real number or symbol variable.

Reference see formula 3-19 and 3-20, and function ***y=exp4_1(w, sita, v).***

3.3 Velocity of a Rigid Body

Define the *spatial velocity* $\hat{V}^s_{ab} \in se(3)$ as

$$\hat{V}^s_{ab} = \dot{g}_{ab} g^{-1}_{ab} \qquad V^s_{ab} = \begin{bmatrix} v^s_{ab} \\ \omega^s_{ab} \end{bmatrix} = \begin{bmatrix} -\dot{R}_{ab} R^T_{ab} p_{ab} + \dot{p}_{ab} \\ (\dot{R}_{ab} R^T_{ab})^\vee \end{bmatrix} \tag{3-21}$$

Define the *body velocity* $\hat{V}^b_{ab} \in se(3)$ as

$$\hat{V}^b_{ab} = g^{-1}_{ab} \dot{g}_{ab} = \begin{bmatrix} R^T_{ab} \dot{R}_{ab} & R^T_{ab} \dot{p}_{ab} \\ 0 & 0 \end{bmatrix} \qquad V^b_{ab} = \begin{bmatrix} v^b_{ab} \\ \omega^b_{ab} \end{bmatrix} = \begin{bmatrix} R^T_{ab} \dot{p}_{ab} \\ (R^T_{ab} \dot{R}_{ab})^\vee \end{bmatrix} \tag{3-22}$$

Then

$$\hat{V}^s_{ab} = \dot{g}_{ab} g^{-1}_{ab} = g_{ab} (g^{-1}_{ab} \dot{g}_{ab}) g^{-1}_{ab} = g_{ab}\, \hat{V}^b_{ab}\, g^{-1}_{ab} \tag{3-23}$$

$$V^s_{ab} = \begin{bmatrix} v^s_{ab} \\ \omega^s_{ab} \end{bmatrix} = \begin{bmatrix} R_{ab} & \hat{p}_{ab} R_{ab} \\ 0 & R_{ab} \end{bmatrix} \begin{bmatrix} v^b_{ab} \\ \omega^b_{ab} \end{bmatrix} \tag{3-24}$$

The 6×6 matrix which transforms twists from one coordinate frame to another is referred to as the *adjoint transformation* associated with *g*, written Ad*g*. Thus, given $g \in SE(3)$ which maps one coordinate system into another, Ad*g* : $\mathbb{R}^6 \to \mathbb{R}^6$ is given as

$$\boxed{\mathrm{Ad}_g = \begin{bmatrix} R & \hat{p}R \\ 0 & R \end{bmatrix}} \tag{3-25}$$

We can easily check that the body and spatial velocities are related by the adjoint transformation,

$$V^s = \mathrm{Ad}_g V^b \tag{3-26}$$

Functions:

function y=Ad(g)

Compute the adjoint matrix of a 4*4 homogenous transformation matrix g, returns a 6*r matrix using formula 3-25

3.4 Forward kinematics and The Manipulator Jacobian

For an n-axis rigid-link manipulator, the *forward kinematic solution* gives the coordinate frame, or pose, of the last link. for revolute joints

$$\xi_i = \begin{bmatrix} -\omega_i \times q_i \\ \omega_i \end{bmatrix} \tag{3-27}$$

For Prismatic joints

$$\xi_i = \begin{bmatrix} v_i \\ 0 \end{bmatrix} \tag{3-28}$$

Then the homogenious transformation matrix of the end effector is

$$g_{st}(\theta) = e^{\hat{\xi}_1\theta_1} \cdots e^{\hat{\xi}_n\theta_n} \cdot g_{st}(0) \tag{3-29}$$

The spatial manipulator Jacobian is defined as:

$$J^{s}_{st}(\theta) = [\xi_1 \quad \xi'_2 \quad \cdots \quad \xi'_n]$$
$$\xi'_i = Ad_{(e^{\hat{\xi}_1\theta_1}\cdots e^{\hat{\xi}_{i-1}\theta_{i-1}})}\xi_i \tag{3-30}$$

Then the spatial velocity of the end-effector is

$$V^s_{st} = J^s_{st}(\theta)\dot{\theta} \tag{3-31}$$

The body manipulator Jacobian is defined as:

$$J^{s}_{st}(\theta) = [\xi^+_1 \quad \xi^+_2 \quad \cdots \quad \xi^+_n]$$
$$\xi^+_i = Ad^{-1}_{(e^{\hat{\xi}_i\theta_i}\cdots e^{\hat{\xi}_n\theta_n} g_{st}(0))}\xi_i \tag{3-32}$$

Then the body velocity of the end-effector is

$$V^b_{st} = J^b_{st}(\theta)\dot{\theta} \tag{3-33}$$

Then the body velocity Jacobian relative to the i-th joint is

$$J^b_{sl_i} = \left[\xi^+_1 \quad \cdots \quad \xi^+_i \quad 0 \quad \cdots \quad 0\right]$$
where
$$\xi^+_j = Ad^{-1}_{(e^{\xi_j\theta_j}\cdots e^{\xi_i\theta_i} gsli0)}\xi_j \tag{3-34}$$

Inverse kinematics problems are usually done by person, we haven't find a efficient way to solve this problem. Peter I. Corke's toolbox solve this problem using a iterative algorithms, his method is time consuming and can just solve numerical inverse problems, not symbol variable.

Functions:

function J=Jacobi(twist,sita)

Compute the spatisl velocity Jacobian relative to the i-th joint using formula 3-34,

where i=pos is a integer, $[\xi_1 \cdots \xi_n]$ =twist is a 6*n twists matrix, n is the number of

joints,[θ1 …θn]=sita is the rotation angles of each twist, gsti0 is the initial homogenous transformation matrix of the i-th joints body coordinates. This function returns a 6*n matrix.

function J=Jacobi_b(twist,sita,gsti0,pos)

Compute the body velocity Jacobian using formula 3-30,where $[\xi_1 \cdots \xi_n]$ =twist is a 6*n twists matrix,n is the number of joints,[θ1 …θn]=sita is the rotation angles of each twist.This function returns a 6*n matrix.

3.5 Manipulator rigid-body dynamics

Let $\theta \in R^n$ be the joint angles for an open-chain manipulator. The Lagrangian of the open-chain manipulator is

$$L(\theta,\dot{\theta}) = \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta} - V(\theta) \qquad 3\text{-}35$$

Using Lagrangian Equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} = \tau_i \qquad 3\text{-}36$$

We can get the formula. (please read the textbook page 219-221)

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + N(\theta,\dot{\theta}) = \tau$$

where

$$M(\theta) = \sum_{i=1}^{n} (J_{sli}^b)^T I_i J_{sli}^b \qquad 3\text{-}37$$

$$C_{ij}(\theta,\dot{\theta}) = \sum_{k=1}^{n} (\Gamma_{ijk})\dot{\theta}_k = \frac{1}{2}\sum_{k=1}^{n} \left(\frac{\partial M_{ij}}{\partial \theta_k} + \frac{\partial M_{ik}}{\partial \theta_j} - \frac{\partial M_{kj}}{\partial \theta_i}\right)\dot{\theta}_k$$

I haven't found an easy way to calculate the forward dynamics, but I find a method to calculate the $M(\theta)$ and Γ_{ijk} .which is also useful for us.

Functions:

function M=dyn_getM(screws,gst0s,sitas,Is)

Compute the matrix $M(\theta)$, screws =[screw1 … screwn] is the 6*n twists matrix , gst0s=[gst01…gst0n] is 4*4n initial position matrix for each joint, sitas=[sita1… sitan] is 1*n rotation angle or translate length, Is=[I1…In] is 6*6n inertia tenser matrix

Reference see formula 3-37.

function dyn_getF(screws,gst0s,sitas,Is)

Compute Γ_{ijk} ,It just print each item of Γ to the screen, does not return any value.

Reference see formula 3-37.

3.6 D-H parameter method for forward kinimatics

There is another popular method for forward kinimatics, which is called Denavit-Hartenberg Parameter method. D-H parameters are obtained by applying a set of rules which specify the position and orientation of frames Li attached to each link of the robot and then constructing the homogenious transformation matrix between frames, denoted by $g_{l_{i-1}l_i}$ .The kinematics of the mechanism can be written as

$$g_{st}(\theta) = g_{l_0 l_1}(\theta_1) g_{l_1 l_2}(\theta_2) \ldots g_{l_{n-1} l_n}(\theta_n) \qquad 3\text{-}35$$

Each of the $g_{l_{i-1}l_i}$ can be get by a series of rotation and transformation, and has the form

T=rot('z,s)*trans(0,0,d)*trans(a,0 ,0)*rot(x,alpha)

```
=[     cos(s), -sin(s)*cos(alpha),    sin(s)*sin(alpha),        cos(s)*a]
 [      sin(s),    cos(s)*cos(alpha), -cos(s)*sin(alpha),           sin(s)*a]
 [      0,         sin(alpha),              cos(alpha),             d]          3-36
 [      0,                  0,                          0,          1]
```

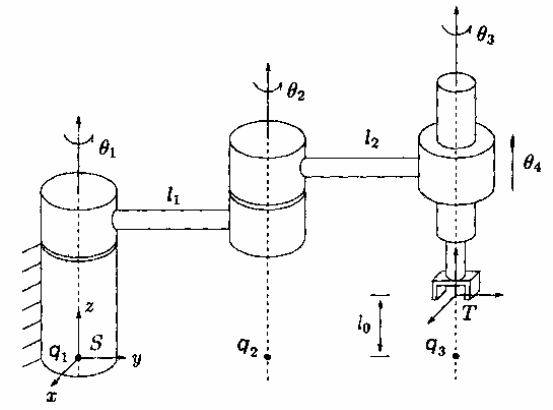
For more information, please read other relative books.

Functions:

function y=DH(sita,d,a,alpha)

Construct the $g_{l_{i-1}l_i}$ in formula 3-36

4 Examples



4.1 Example 3.1 (Page 115 of the book). SCARA forward kinematics

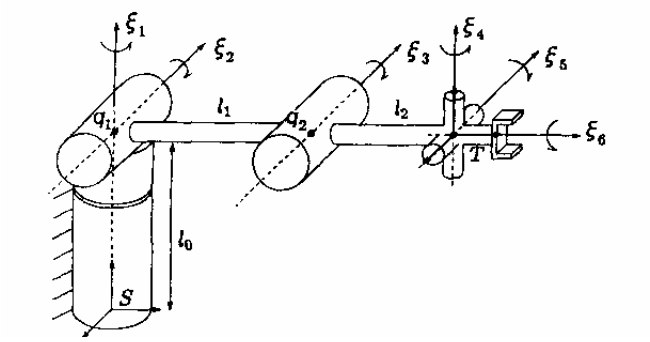
[S]:

```
syms l1 l2 s1 s2 s3 s4 l0
R1=exp4_2([0; 0; 0; 0; 0; 1],s1)
R2=exp4_2([l1; 0; 0; 0; 0; 1],s2)
R3=exp4_2([l1+l2; 0; 0; 0 ;0 ;1],s3)
R4=exp4_2([0; 0; 1; 0; 0; 0],s4)
gst0=[eye(3)    [0;l1+l2;l0];
          0 0 0        1]
Rst=R1*R2*R3*R4*gst0
Rst =simple(Rst)
```

[result]

```
Rst =
  [cos(s1+s2+s3), -sin(s1+s2+s3), 0, -sin(s1+s2)*l2-sin(s1)*l1]
[sin(s1+s2+s3), cos(s1+s2+s3), 0,    cos(s1+s2)*l2+cos(s1)*l1]
[0,      0,              1,                                  l0+s4]
[0,    0,      0,                                          1]
```

4.2 Example 3.2(Page 117 of the book) Elbow manipulator kinematics



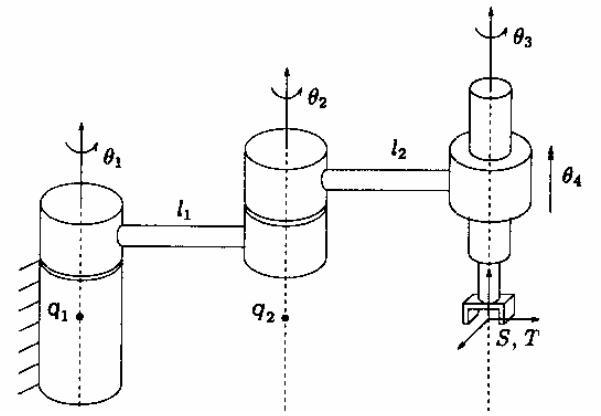
[S]:

```
syms l0 l1 l2 s1 s2 s3 s4 s5 s6
w1=[0 0 1];w2=[-1 0 0 ];w3=w2;
w4=w1;w5=w2;w6=[0 1 0];
q1=[0 0 l0];
screw1=screw(w1,q1)
screw2=screw(w2,[0 0 l0])
screw3=screw(w3,[0 l1 l0])
screw4=screw(w4,[0 l1+l2 l0])
screw5=screw([-1 0 0 ],[0 l1+l2 l0])
screw6=screw(w6,[0 l1+l2 l0])
R1=exp4_2(screw1,s1)
R2=exp4_2(screw2,s2)
R3=exp4_2(screw3,s3)
R4=exp4_2(screw4,s4)
R5=exp4_2(screw5,s5)
R6=exp4_2(screw6,s6)
gst0=[eye(3,3) [0 ;l1+l2; l0] ;
0 0 0 1]
Rst=R1*R2*R3*R4*R5*R6
gst=Rst*gst0
gst=simple(gst)
```

[result]

R1 to R6 and gst is too long,so I omit them.

4.3 Example 3.3(Page 120 of the book) SCARA forward kinematics with alternate base frame

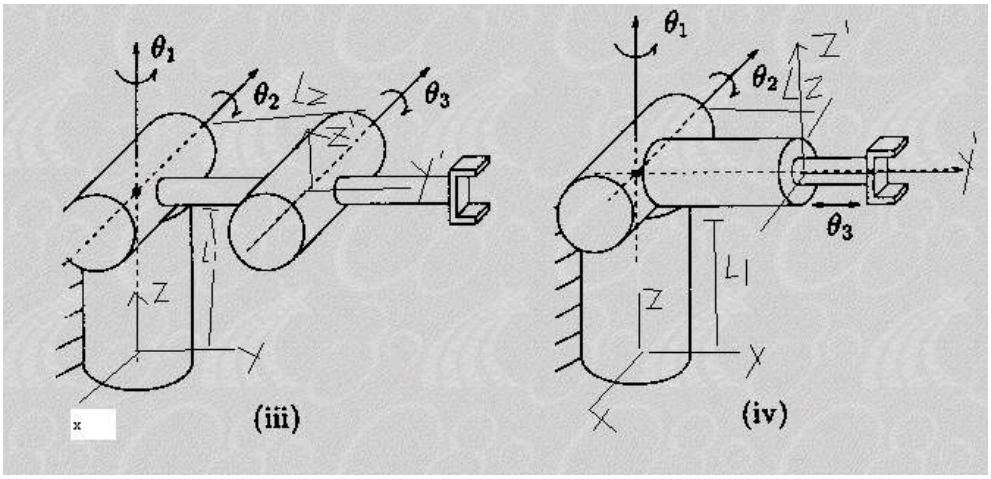


[S]

```
syms l0 l1 l2 s1 s2 s3 s4 s5 s6
screw1=screw([ 0 ;0; 1],[0 ;-l1-l2;0])
screw2=screw([ 0 ;0 ;1],[0;-l2;0])
screw3=screw([ 0 ;0; 1],[0;0;0])
screw4=[0 ;0; 1;0;0;0 ]
R1=exp4_2(screw1,s1)
R2=exp4_2(screw2,s2)
R3=exp4_2(screw3,s3)
R4=exp4_2(screw4,s4)
gst0=eye(4)
gst=R1*R2*R3*R4*gst0
simple(gst)
```

[result is omitted here]

4.4 Homework 3.3(Page 190 of the book) forward kinematics



[S]

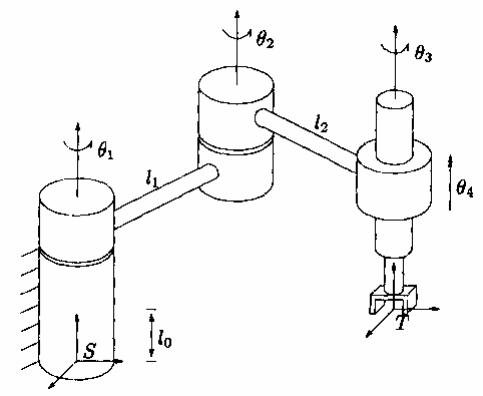
(iii)

```
syms l0 l1 l2 s1 s2 s3 s4 s5 s6
screw1=screw([ 0 ;0; 1],[0 ;0;l1])
screw2=screw([ -1;0 ;0],[0;0;l1])
screw3=screw([ -1;0 ;0],[0;l2;l1])
R1=exp4_2(screw1,s1)
R2=exp4_2(screw2,s2)
R3=exp4_2(screw3,s3)
gst0=[eye(3) [0;l2;l1];
0 0 0 1]
gst=R1*R2*R3*gst0
simple(gst)
```

(iv)

```
syms l0 l1 l2 s1 s2 s3 s4 s5 s6
screw1=screw([ 0 ;0; 1],[0 ;0;l1])
screw2=screw([ -1;0 ;0],[0;0;l1])
screw3=[0 1 0 0 0 0]'
R1=exp4_2(screw1,s1)
R2=exp4_2(screw2,s2)
R3=exp4_2(screw3,s3)
gst0=[eye(3) [0;l2;l1];
0 0 0 1]
gst=R1*R2*R3*gst0
J=jacobi([screw1 screw2 screw3],[s1 s2 s3])
simple(gst)
```

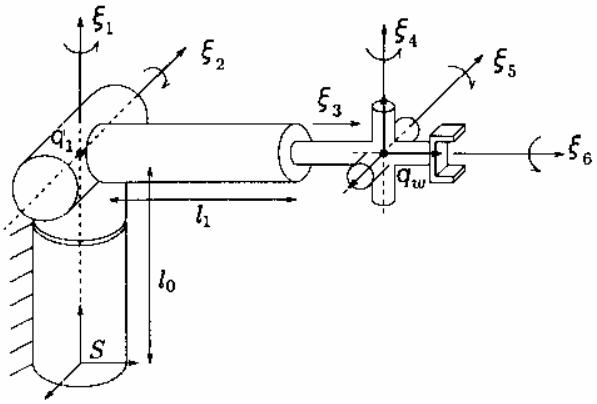
4.5 Example 3.8(page 154 of the book) Jacobian for SCARA robot



[S]:

```
clc
clear
syms l0 l1 l2 s1 s2 s3 s4 s5 s6
screw1=screw([ 0 ;0; 1],[0 ;0;0])
screw2=screw([ 0 ;0 ;1],[0;l1;0])
screw3=screw([ 0 ;0; 1],[0;l1+l2;0])
screw4=[0 ;0; 1;0;0;0 ]
J=Jacobi([screw1 screw2 screw3 screw4],[s1 s2 s3 s4])
simple(J)
```

4.6 Example 3.9 Jacobian for Stanford Hand

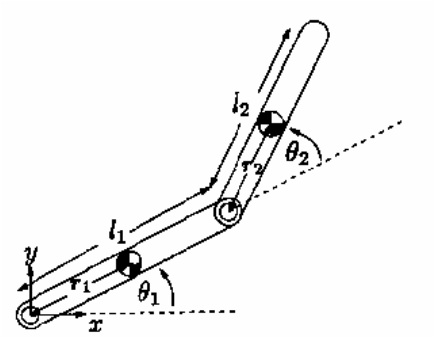


[S]:

```
clc
clear
syms l0 l1 l2 s1 s2 s3 s4 s5 s6 l3 l4 l5 l6
screw1=screw([ 0 ;0; 1],[0 ;0;0])
screw2=screw([ -1 ;0 ;0],[0;0;l0])
screw3=[0;1;0;0;0;0]
screw4=screw([ 0 ;0; 1],[0;l1+s3;l0]) %right or wrong??
```

```
screw5=screw([ -1 ;0; 0],[0;l1+s3;l0])
%screw4=screw([ 0 ;0; 1],[0;l1;l0]) %right or wrong??
%screw5=screw([ -1 ;0; 0],[0;l1;l0])
screw6=screw([ 0 ;1; 0],[0;l1+s3;l0])
J=Jacobi([screw1 screw2 screw3 screw4 screw5 screw6],[s1 s2 s3 s4 s5 s6])
simple(J)
```

4.7 Example (Page 212 of the book) Dynamics of a two link plannar robot



[S]:

```
syms l0 l1 l2 s1 s2 s3 s4 s5 s6 r0 r1 r2 r3 m1 m2 m3 Ix1 Ix2 Ix3 Iy1 Iy2 Iy3 Iz1 Iz2 Iz3

screw1=[0 0 0 0 0 1]'
screw2=[0; -l1; 0; 0; 0; 1]

gst10=[eye(3,3) [r1 ;0;0] ;
0 0 0 1]
gst20=[eye(3,3) [l1+r2 ;0;0] ;
0 0 0 1]

M1=diag([m1 m1 m1 Ix1 Iy1 Iz1])
M2=diag([m2 m2 m2 Ix2 Iy2 Iz2])

%M=simple(dyn_getM([screw1 screw2 screw3],[gst10 gst20 gst30],[s1 s2 s3],[M1 M2 M3]))
dyn_getF([screw1 screw2],[gst10 gst20],[s1 s2 ],[M1 M2])
```

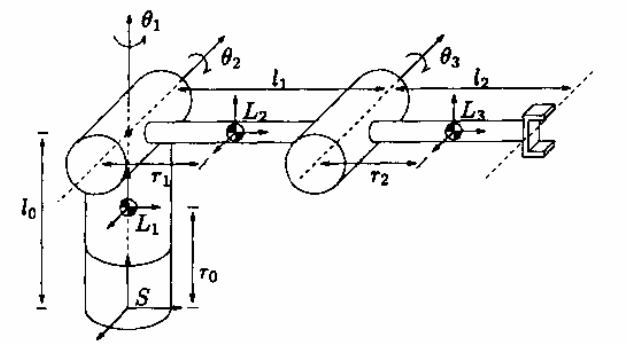
[result]:

```
M =
[2*r2*m2*l1*cos(s2)+Iz1+m2*l1^2+r1^2*m1+Iz2+r2^2*m2,    r2*m2*l1*cos(s2)+r2^2*m2+Iz2]
[r2*m2*l1*cos(s2)+r2^2*m2+Iz2,                          r2^2*m2+Iz2]

F111: 0
F112:-r2*m2*l1*sin(s2)
F121:-r2*m2*l1*sin(s2)
```

```
F122:-r2*m2*l1*sin(s2)
F211:r2*m2*l1*sin(s2)
F212:0
F221:0
F222:0
```

4.8 Example 4.3(Page 222 of the book) Dynamics of a three link manipulator



[S]:

```
syms l0 l1 l2 s1 s2 s3 s4 s5 s6 r0 r1 r2 r3 m1 m2 m3 Ix1 Ix2 Ix3 Iy1 Iy2 Iy3 Iz1 Iz2 Iz3

screw1=[0 0 0 0 0 1]'
screw2=[0; -l0; 0; -1; 0; 0]
screw3=[0; -l0; l1; -1; 0; 0]

gst10=[eye(3,3) [0 ;0;r0] ;
0 0 0 1]
gst20=[eye(3,3) [0 ;r1;l0] ;
0 0 0 1]
gst30=[eye(3,3) [0 ;l1+r2;l0] ;
0 0 0 1]

M1=diag([m1 m1 m1 Ix1 Iy1 Iz1])
M2=diag([m2 m2 m2 Ix2 Iy2 Iz2])
M3=diag([m3 m3 m3 Ix3 Iy3 Iz3])

%M=simple(dyn_getM([screw1 screw2 screw3],[gst10 gst20 gst30],[s1 s2 s3],[M1 M2 M3]))
dyn_getF([screw1 screw2 screw3],[gst10 gst20 gst30],[s1 s2 s3],[M1 M2 M3])
```

5 Future Work

We have finished the algorithms about twist, forward kinimatics, manipulator forward Dynamics ,and manipulator Jacobian. You can use this toolbox to solve most problems about these relative questions, either numerical or symbol, and all our work is based on MATLAB 6.5.
There are still something not finished, such as automatic inverse kinimatics, inverse dynamics,and trajectory generation. They are very useful,but we have many difficulties solving them. If you have some good ideas or algorithms, please tell us. You can contact us by email.

6 Appendix : ToolBox Function List

function M=Rot(type,sita)
Generate a Rotation matrix, type=1 2 3 represent for the rotation axis x, y, z respectively. sita represent for the rotation angle.
In fact, this function generate a 3*3 matrix M as follows:

function M=Trans(dx,dy,dz)
Generate a translation matrix, dx, dy, dz represent for the translation along x, y, z axis respectively.

function y=VectorProduct(x1,x2)
Compute the vector product of two 3*1 vector x1,x2,return a 3*1 vector
Reference See formula 3-1

function y=v3tor3(v)
Compute the skew symmetry matrix of a 3*1 vector v
Reference See formula 3-2s

function y=exp3_1(omiga,sita)
Compute the exponential product. Using formula

$$e^{\omega^{\wedge}\theta} = I + w^{\wedge}\sin(\omega) + w^{\wedge 2}(1-\cos(\omega))$$

Note: omega is a 3*1 unit vector, sita is a real number denoting the angle of rotation.

This function returns a 3 by 3 matrix whose value is the exponential product $e^{\omega^{\wedge}\theta}$.

function y=screw(w,q)
Compute the pure rotation twist--a 6 by 1 vector--which rotates about the axis w(a

3*1 unit vector), and across the point q(a 3*1 vector).The function is useful in forward kinematics.

function y=exp4_1(w, sita, v)

Compute the exponential product of a twist, the twist is [v, w]',where w is a 3*1 unit vector, and v is a 3*1 vector, sita is the rotation angle, which is a real number or symbol variable.

Reference see formula 3-19 and 3-20.

function y=exp4_2(vw,sita)

Compute the exponential product of a twist vw, a 6 by 1 vector, and the rotation angle sita, which is a real number or symbol variable.

Reference see formula 3-19 and 3-20,and function ***y=exp4_1(w,sita,v).***

function y=Ad(g)

Compute the adjoint matrix of a 4*4 homogenous transformation matrix g, returns a 6*r matrix using formula 3-25

function J=Jacobi(twist,sita)

Compute the spatial velocity Jacobian relative to the i-th joint using formula 3-34,where i=pos is a integer, $[\xi_1 \cdots \xi_n]$ =twist is a 6*n twists matrix, n is the number of joints,[θ1 …θn]=sita is the rotation angles of each twist, gsti0 is the initial homogenous transformation matrix of the i-th joints body coordinates. This function returns a 6*n matrix.

function J=Jacobi_b(twist,sita,gsti0,pos)

Compute the body velocity Jacobian using formula 3-30, where $[\xi_1 \cdots \xi_n]$ =twist is a 6*n twists matrix, n is the number of joints,[θ1 …θn]=sita is the rotation angles of each twist. This function returns a 6*n matrix.

function y=DH(sita,d,a,alpha)

Construct the $g_{l_{i-1}l_i}$ in formula 3-36

function M=dyn_getM(screws,gst0s,sitas,Is)

Compute the matrix $M(\theta)$, screws =[screw1 … screwn] is the 6*n twists matrix ,gst0s=[gst01…gst0n] is 4*4n initial position matrix for each joint,sitas=[sita1… sitan] is 1*n rotation angle or translate length, Is=[I1…In] is 6*6n inertia tenser matrix

Reference see formula 3-37.

function dyn_getF(screws,gst0s,sitas,Is)

Compute Γ_{ijk} , It just print each item of Γ to the screen, dose not return any value.

Reference see formula 3-37.